

The Solution to the Alice and Bob Paradox: Pseudoprivate Key Encryption

19 June 2024

Simon Edwards

Research Acceleration Initiative

Introduction

For several decades, researchers have sought a solution to the fundamental logical paradox posed by a need to inform a distant party of the parameters of a key, the need for the key to be known by the distant party, as well as the need to ensure that an eavesdropper, despite their being able to intercept all communicated information, cannot infer the key whilst continuing to ensure that the intended party may make the necessary inferences.

As the need for robust cryptography has never been more apparent for reason of increasingly speedy computers, the increased probability of cyberattacks by nation-state entities, increased societal reliance on E-Commerce and the need for personal privacy in the face of unprecedented governmental and corporate surveillance of civilians, a system which may enable highly secure conversations between distant parties would solve a great many problems.

Abstract

Rather than encapsulating private keys within obfuscatory mechanisms (currently referred to as Key Encapsulation Modules in the industry,) I propose a much simpler approach. This approach is based upon the concept (never before promulgated by any party) of sending byte values which represent a range of possible true key values rather than transmitting the actual key value in any form, no matter how obfuscated.

It is only by not transmitting precise information concerning the private key that security may be ensured. In this regime, Bob sends Alice a byte value which defines the low end of a range of *three possible values*. If Bob sends, for example, hex 05, then the true value may be 05, 06, or 07. This is step one.

Alice then responds by sending a two-byte response. The first byte of this response may represent one of two possible queries. The second byte would be used, in this regime, to define which of the three possible true values are the subject of the query. The eavesdropper does not know which of these two queries are represented by which byte value and does not know which byte value refers to possible true value A, possible true value B and possible true value C. These definitions are pre-set in the closed source code of the module and are unknown even to the operators of the modules.

Alice responds to Bob's message by asking one of two possible questions: Is possible true value A the correct value? OR Is possible true value A NOT the correct value?

In response, Bob must answer all of Alice's questions truthfully in what we might term *condition one*. *Condition one* is the condition under which the pair communicate prior to the successful inference of the true value of any given private key byte. Bob might be asked by Alice if byte A is the correct value and he might respond in the negative. If he does, Bob will be asked by Alice, at random, whether B is the value, whether B is NOT the value, whether C is the value, or whether C is NOT the value. Bear in mind, the eavesdropper does not know which of two values is meant to represent query A and which is meant to represent query B. The eavesdropper does not know which of two byte values is being used by the pair to represent an affirmative or negative response. If the range of possible values were merely two, the eavesdropper would be able to cycle through, in the case of a 1024-byte key, 4096 possible keys with ease. However, when there are three or more possible true values in the undefined range, this paradigm is turned on its head.

Enter *condition 2*. When there are three possible true values in the range, there is a 1 in 6 chance that Alice will both choose to ask query A and will guess correctly on the first try. When this occurs, Alice will know, given that Bob answers truthfully (and he must) that the pair are now in *condition 2*. Alice knows she has a right answer and Bob knows he's provided it. He also knows that the eavesdropper cannot know that this is what transpired, particularly if Alice does the following to fulfill her *condition 2* responsibilities:

Alice, knowing she already has the correct value for byte 1 of the key, purposefully and deceitfully asks Bob more questions. She asks Bob about the other possible byte values and treats his "yes" as if it is a "no."

Alice and Bob have agreed ahead of time to allow for the true value of the first byte to be used to permute the very language which they speak. They know that even using this system, without this additional step, an eavesdropper may eventually be able to infer which value represents yes and which no and which represents A, B, and C if they had enough data to analyze. If they change their language, simplistic though it may be, prior to the eavesdropper gaining enough information to make any conclusive inference as to the beginning state of the "language," the eavesdropper would remain perpetually one step behind. This shifting is as simple as taking the true hex value once it is inferred and exponentializing it against a predefined prime exponent and then multiplying that value against the previous totem (exponentialized by a different prime value) and then truncating that value in order to define the new totem.

The eavesdropper may guess at the first byte value and have a one in three chance of guessing correctly, but his guess gets him no closer to learning the meaning of the totem language being used by Alice and Bob.

Borrowing from the concept of escrow, the software module used to support this would be, itself, closed source but with multiple authors being used to set the crucial parameters of initial totem values as well as prime values. The system

would rely on a secret initial hex value to represent query A, query B, and possible values A, B, and C. The two prime exponents used to redefine totem values would each be set by additional third parties during software development.

If, somehow, the security of the system were compromised due to dishonesty on the part of all involved in software development, it would quickly become apparent as the variables would become public knowledge and the compromise could be verified by the user community. At that point, a new system could be implemented based upon similar principles but in which different escrow knowledge holders are entrusted.

Conclusion

This may be considered to be the first concept for a public-private key hybrid. The mechanism used for the actual cryptography would be ordinary, run-of-the-mill symmetric key AES or some other private key scheme, but the dissemination of peri-truthful information concerning the key technically means that the key has some features of a public key i.e. if all of the sensitive values used in the software development were leaked, the system would become useless. The advantage of this system is that it would enable all Internet users' privacy to be wholly protected (so long as they protect their private keys) by the continued secrecy of those specialized values upon which the basic communications protocol is based.

So-called "quantum safe" requirements would be met by this system as extremely long key lengths could be supported through this sort of back-and-forth dialogue between users. It would require only 4096 such brief exchanges between nodes in order to establish what may be termed from heretofore as a *pseudoprivate key*. Like a pseudorandom number generator, the numbers may not be truly random, but this is a distinction without a difference in both cases.